基于改进的鸡群算法在云计算资源调度中的研究 *

陈　暄[1]，龙　丹[2]

(1. 浙江工业职业技术学院, 浙江 绍兴 312000; 2. 浙江大学, 杭州 310058)

摘　要：针对云计算中的资源调度效率低的问题，提出将改进后的鸡群算法用于调度。首先，引入反向学习概念对鸡群种群进行初始化，提高全局搜索能力；其次，对小鸡的位置引入了粒子群算法中的权重值和学习因子的概念进行改进，优化了鸡群个体位置；再次通过差分算法对鸡群算法整体的个体位置进行优化，最后通过边界处理从整体上预防了算法中个体位置可能出现的越界。在仿真实验中，将优化后的鸡群算法与基本鸡群算法，粒子群算法和蚁群算法进行在完成时间，花费成本、能量消耗和负载均衡中进行了对比，取得了较好的效果。

关键词：鸡群算法 反向学习 学习因子 差分算法

中图分类号：TP301　　doi: 10.3969/j.issn.1001-3695.2018.03.0161

Based on improved chicken swarm optimization in cloud computing resource scheduling

Chen Xuan[1], Long Dan[2]

(1. *Zhejiang Industry Polytechnic College*, *Shaoxing Zhejiang* 312000, *China*; 2. *Zhejiang University*, *Hangzhou* 310058, *China*)

Abstract: In order to solve the problem of low efficiency of resource scheduling in cloud computing, it is proposed to schedule the improved chicken swarm optimization. First, the concept of reverse learning is used to initialize the chicken swarm and improve the global search capability. Secondly, the position of chick is introduced into the concept of weight value and learning factor in particle swarm optimization to improve the individual position of the flock; the individual position of the chicken swarm optimization is again optimized by the difference algorithm, and finally the whole is processed by boundary processing to prevent possible cross-border of individual locations in the optimization. In the simulation experiment, the optimized chicken swarm optimization and the basic chicken swarm optimization, the particle swarm optimization algorithm and the ant colony algorithm are compared in terms of completion time, cost, energy consumption and load balance, and good results have been achieved.

Key words: chicken swarm optimization; reverse learning; learning factor; difference algorithm

0　引言

如何能够进行高效的资源调度是目前云计算研究的重点[1],众多研究学者将遗传算法[2,3]、粒子群算法[4,5]、蚁群算法[6,7]等多种智能算法运用在资源调度中取得了较好的效果。Meng 在 2014 年提出了鸡群算法(chicken swarm optimization,CSO)[8],该算法是一种基于群体的随机优化算法,该算法的优点是实现简单,缺点是容易陷入局部最优而无法获得最优解且存在收敛速度慢,精度低等问题。针对鸡群算法的运用,学者们进行了一些研究,文献[9]提出将改进的鸡群算法运用在微型电网指标优化的过程中,结果说明取得了较好的效果,存在的不足是在实验部分仅仅与粒子群算法进行了对比,缺乏与基本的鸡群算法对比,优化结果说服力稍显不足;文献[10,11]将改进的鸡群算法运用到车间作业的调度问题中,从整体上提高了调度的效率,获得了比较好的效果,存在的不足是缺乏与经典的调度算法进行对比;文献[12]鸡群算法中的公鸡和母鸡分别进行改进,并用于多分类器系数的优化,仿真实验通过对 6 种数据集分别进行了分类优化,取得了较好的分类效果,存在的不足是缺乏与常用的多分类器算法对比;文献[13,14]将鸡群算法运用到水利调度、家电负荷分解中,获得了一定的效果。

本文在以上研究的基础上,在云计算资源调度中采用改进的鸡群算法(improve chicken swarm optimization,ICSO)对其资源进行有效的分配,通过对鸡群算法种群初始化,更新小鸡位置,差分算法选择最优体和边界处理提高了算法在局部和全局的搜索能力,在云计算的仿真实验结果中,本文算法与几种经典智能算法对比,具有一定的优势,能够有效的提高云计算资源调度的效果。

收稿日期：2018-03-14；**修回日期：**2018-04-18　**基金项目：**国家自然科学基金资助项目（LQ18A010003，11426205）；绍兴市科技局项目（2015B70013）

作者简介：陈暄（1979-），男，江西人，副教授，硕士，主要研究方向为云计算、无线传感（chenxuan1979@sina.com）；龙丹（1975-）男，湖南人，讲师，博士，主要研究方向为图像处理和分析、算法设计.

1　云计算资源调度描述

云计算中的任务分配资源是一个完全 NP 问题,本文从用户的需求和资源拥有者的角度来解决资源分配问题,通过引入时间、花费和能耗的限制,提出云计算资源调度模型。它能够满足用户的 QoS 需求,能够有效的实现资源的负载均衡。设定有 n 个任务共享 m 个资源,每一个任务 S_i 通过 $k(i)$ 个并行且相互依赖的子任务组成,设定这些子任务具有相同的计算量,每一个计算资源 R_j 都有一个固定的价格 P_j 和能耗 M_j 。当多个子任务被分配到 R_j 上的时候,子任务就会等比例的分享 R_j 的计算能力。而资源分配其核心的问题是解决一个 $n\times m$ 的矩阵 a ,行代表任务,列代表资源, $a_{i,j}$ 表示任务 S_i 分配到资源 R_j 的子任务个数, a_i 表示矩阵 a 第 i 行,满足 $\sum_{a_{ij}\in a_i} a_{ij}=k(i)$,同样的原理可以得到时间、花费和能耗的三个矩阵,分别记为 T , E 和 U ,矩阵 T 中的元素 t_{ij} 表示资源 R_j 完成任务 S_i 分配到它上面的 a_{ij} 个子任务上所花费的时间,由于子任务都是并行完成的,因此任务 S_i 完成的时间是 $\max\{t_{ij}\in t_i\}$,矩阵 E 中的 e_{ij} 表示资源 R_j 完成任务 S_i 分配到它上面的 a_{ij} 个子任务上所花费,因此任务 S_i 的花费是 $\sum_{j=1}^{m} e_{ij}$,矩阵 U 中的 u_{ij} 表示资源 R_j 完成任务 S_i 分配到它上面的 $a_{i,j}$ 个子任务上能量消耗,因此任务 S_i 的能量消耗为 $\sum_{j=1}^{m} u_{ij}$ 。通常在任务相关指标中设定权重值保持三个变量之间的权衡关系,使用 ω_t , ω_e 和 ω_u 来表示,且 $\omega_t+\omega_e+\omega_u=1$,因此得到任务 S_i 的总的花费是 $\omega_t\bullet\max_{t_{ij}\in t_i}\{t_{ij}\}+\omega_e\bullet\sum_{j=1}^{m}e_{ij}+\omega_u\bullet\sum_{j=1}^{m}u_{ij}$,使用公式 $u_i(a_i)$ 表示效用,即

$$u_i(a_i)=\frac{1}{\omega_t\bullet\max_{t_{ij}\in t_i}\{t_{ij}\}+\omega_e\bullet\sum_{j=1}^{m}e_{ij}+\omega_u\bullet\sum_{j=1}^{m}u_{ij}}\tag{1}$$

因此每一个任务的目标就是将它的效用最大化,即任务 S_i 中的花费、时间和能耗最小化,因此任务总效用如下:

$$u(a)=\sum u_i(a_i)\tag{2}$$

因此,云计算的任务资源调度效用的最佳值就是全部任务效用的最好值。

2　鸡群算法

鸡群算法主要是将优化问题描述成鸡群在搜寻食物的过程, 主要是模拟鸡群中存在的等级分类和觅食方式。算法思想是将整个鸡群分为若干个子群, 每一个子群由一只公鸡、一定数量的母鸡和一定数量的小鸡构成, 由于具有不同的等级制度, 因此不同的子鸡群之间存在一定的竞争，通过竞争获得最佳位置的鸡群个体。具体过程如下:

a) 整个鸡群中存在若干个子群, 每一个子群包含了一个公鸡, 若干个母鸡和小鸡;

b) 鸡群算法中将适应度值最好的若干个鸡称为公鸡并分布在不同的子群中, 将适应度值最差的鸡作为小鸡, 除去以上两种适应度值外的鸡作为母鸡, 母鸡具有随意选择子群和对应小鸡的权利，同时母鸡和小鸡之间的关系是随机分配的;

c) 在鸡群中的等级分类以及支配关系只有在更新的时候才发生改变，即母鸡与小鸡之间这种关系一旦建立就保持不变直到更新;

d) 子群中的个体都必须以所在群的公鸡为中心进行寻找食物, 并时刻防止其他子群中鸡群个体进入本子群抢夺食物, 子群中的小鸡具备抢夺其他子群中食物的功能并且在寻找食物的时候伴随着母鸡一起寻找。鸡群中存在部分个体在获取食物的地位上具有一定的优先权，能够在地位竞争中占据一定的主动。

鸡群中的个体都按照自己的规律运动, 直到找到最佳的位置, 因此将鸡群中的个体位置对应优化问题的一个解, 找到位置最优的个体即为优化问题的最优解, 在整个鸡群算法中, 设定所有鸡群的个体数目为 N , 每个鸡群个体所在位置采用 $x_{i,j}(t)$ 表示, 其意义表示在第 i 个鸡群个体在第 j 维中的第 t 次迭代中所得到的位置。因此，对于鸡群算法中的三种不同类型的鸡具有不同的位置, 即鸡群个体的位置更新伴随着鸡的种类不同而采用不同的位置变换。公鸡在每个子群中具有最好的适应度值, 它能够在广泛的空间中寻找并定位食物。

公鸡对应的位置更新如下:

$$x_{i,j}(t+1)=x_{i,j}(t)\bullet(1+Randn(0,\sigma^2))\tag{3}$$

$$\sigma^2=\begin{cases}1 & if\ \ f_i\le f_k\\ \exp(\dfrac{f_k-f_i}{|f_i|+\varepsilon}) & otherwise\end{cases}\tag{4}$$

其中： $Randn(0,\sigma^2)$ 为均值为 0, σ^2 对应的是一个高斯分布, ε 表示为一个很小的常数, k 表示另一个公鸡个体。

母鸡的位置更新如下:

$$\begin{aligned}x_{i,j}(t+1)=x_{i,j}(t)+c_1\bullet rand\bullet(x_{r1,j}(t)-x_{i,j}(t))\\+c_2\bullet rand\bullet(x_{r2,j}(t)-x_{i,j}(t))\end{aligned}\tag{5}$$

$$c_1=\exp((f_i-f_{r1})/abs(f_i)+\varepsilon)\tag{6}$$

$$c_2=\exp(f_{r2}-f_i)\tag{7}$$

其中： $rand$ 为[0, 1]的均匀分布的随机数, r_1 表示第 i 只母鸡自身所在子群中公鸡个体, r_2 表示整个鸡群中在公鸡和母鸡中随机选取的任意个体, 其这两者不同即 $r_1\ne r_2$ 。

小鸡的位置更新如下:

$$x_{i,j}(t+1)=x_{i,j}(t)+F\bullet(x_{m,j}(t)-x_{i,j}(t))\tag{8}$$

其中： m 表示第 i 只小鸡对应的母鸡, F 为跟随系数, 表示小鸡跟随母鸡寻找食物。

3　改进的鸡群算法在云计算中的调度设计

3.1　初始化选择

CSO 算法中描述的种群是没有初始化的,导致的后果就是

无法保证算法最优解能够均匀分布在空间中,从而限制了算法的求解效率,降低了算法的性能。反向学习[15]是一种运用在机器学习的优化策略,在算法的每次迭代中,得到这些当前解的所有反向解,并在当前解与反向解之间选择出有利于进化的解,降低算法的盲目性,本文采用优化的反向学习策略[16]来扩大整个种群的初始化搜索范围,这样能够进一步的扩大算法的全局搜索能力，避免陷入局部最优。算法过程如下:

a) 随机选取 CSO 种群 $NP_1=\{x_1(t),x_2(t),\cdots x_N(t)\}$，

$$x_i(t)=(x_{i,1}(t),x_{i,2}(t),\cdots x_{i,j}(t),\cdots x_{i,D}(t)),\ i\in N \tag{9}$$

b) 求与 NP_1 对应的反向种群 $NP_{op}=\{\tilde{x}_1(t),\tilde{x}_2(t),\cdots\tilde{x}_p(t)\}$，并按照文献[16]求出每一个体

$$\tilde{x}_i(t)=(\tilde{x}_{i,1}(t),\tilde{x}_{i,2}(t),\cdots\tilde{x}_{i,j}(t),\cdots\tilde{x}_{i,D}(t)) \tag{10}$$

c) 从 $NP_1\cup NP_{op}$ 中选择适应度值最好的个体 x_{best}，计算 $x_{mean}=(x_1+x_2+\cdots x_{2N})/2N$，通过式(11)得到。

$$x_{opbest}=\begin{cases}x_{best},f(x_{best})<f(x_{mean})\\x_{mean},otherwise\end{cases} \tag{11}$$

通过反向学习的方法使得算法在更大的搜索空间中搜寻到最优值,能够引导个体向着最优值方向进化,从而提高了整体算法的收敛速度。

3.2　小鸡位置更新

CSO 算法中,小鸡的位置的更新仅仅只是跟随母鸡的位置移动,而并没有向算法中的最优适应个体的公鸡方向的移动,这样在一定程度上容易导致个体陷入局部最优从而使得算法的整体效率降低,CSO 中的小鸡和粒子群算法中的粒子具有相似的地方,粒子群算法中的粒子在局部获得位置得和全局获得最优解位置就相当于小鸡在自身母鸡旁边获得局部位置以及在整个公鸡引导的群中的最优位置。因此本文借助粒子群算法中的学习因子的概念,对式(8)进行更新:

$$\begin{aligned}x_{i,j}(t+1)=\omega\bullet x_{i,j}(t)+\lambda1\bullet(x_{m,j}(t)\\-x_{i,j}(t))+\lambda2\bullet(x_{r,j}(t)-x_{i,j}(t))\end{aligned} \tag{12}$$

其中：m 表示子群中小鸡跟随对应的母鸡的个体，r 为子群中的小鸡对应的公鸡的个体，$\lambda1$ 和 $\lambda2$ 分别表示两个学习因子，即 $\lambda1$ 表示小鸡向母鸡学习的程度因子，$\lambda2$ 表示小鸡向公鸡学习的程度因子，ω 表示权重。

3.2.1 权重值设定

权重值的大小的设定关系到后期算法中的种群多样性的问题,因此本文采用一种非线性动态惯性权重[17]的表达方式:

$$\omega=1-\left(\omega_{max}\bullet e^{\left(-\left(\frac{t-1}{T_{\max}}\right)\right)}+\frac{k}{2}\bullet\omega_{\min}\bullet\lg(t)-\left|\frac{T_{\max}-t}{p\bullet T_{\max}}\right|\right) \tag{13}$$

其中：ω_{max} 和 $\omega_{\min}$ 分别表示惯性权重的最大值和最小值，$T_{\max}$ 为最大迭代次数，k,p 为控制因子，对权重进行调节作用，这样能够保证小鸡个体在算法前期的所在群中进行全局搜索，从而提高算法的精度，而在后期加强局部搜索，提高整体算法的解的整体精度。

3.2.2 设定学习因子

$\lambda1$，$\lambda2$ 因子的值在一定程度上能够决定小鸡向母鸡学习和向公鸡学习的程度，当 $\lambda1,\lambda2$ 值比较小的时候，允许小鸡能够在当前群中寻找到更好的解，当 $\lambda1,\lambda2$ 的值比较大的时候，小鸡有较大的搜索空间，使得小鸡能够搜索其余空间中的解。显然 $\lambda1,\lambda2$ 的值的设定容易导致算法过早陷入收敛或者发散的状态，因此需要对 $\lambda1,\lambda2$ 的值进行限制，设定 $\lambda_{\min}$ 和 $\lambda_{\max}$ 分别表示为学习因子的最大值和最小值，$t_{current}$ 为当前迭代次数，$t_{\max}$ 和 $t_{\min}$ 分别表示设定的最大，最小次数，这样设置的目的使得小鸡能够更好的向公鸡、母鸡学习。

$$\lambda_1=\begin{cases}\lambda_1\times\dfrac{t_{\max}-t_{current}}{t_{\max}},\lambda_1>\lambda_{\min}\\\lambda_{\min}\qquad\qquad\lambda_1\le\lambda_{\min}\end{cases} \tag{14}$$

$$\lambda_2=\begin{cases}\lambda_2\times\dfrac{t_{\max}+t_{current}}{t_{\max}},\lambda_2<\lambda_{\max}\\\lambda_{\max}\qquad\qquad\lambda_2\ge\lambda_{\max}\end{cases} \tag{15}$$

3.3　引入差分算法选择最优个体

针对 CSO 算法中的的最优位置个体的选择，本文选择了差分算法。它是一种启发式随机搜索算法，对种群中的个体进行求解，其过程包含变异，交互和选择三个部分构成。

a) 变异操作。设 $x_{i,j}^t$ 表示在种群第 t 代迭代中的第 i 个 j 维的个体，依据式(16)进行变异操作。

$$V_{i,j}^{t+1}=x_{i,j}^t+F*(x_{r1,j}^t-x_{r2,j}^t) \tag{16}$$

其中：$V_{i,j}^{t+1}$ 即为变异后的个体，F 为[0, 1]的随机因子，其作用是控制保持差分向量的伸缩度。

b) 交叉操作。当概率的值为[0, 1]时，将第 j 维中的第 i 个体 $x_{i,j}^t$ 与变异后的个体 $V_{i,j}^{t+1}$ 进行交叉，得到新的个体；否则还是原值。

$$\kappa_{i,j}^t=\begin{cases}V_{i,j}^{t+1},\quad p\in[0,1]\\x_{i,j}^t,otherwise\end{cases} \tag{17}$$

c) 选择操作。在选择操作过程中，主要采用“贪婪”策略进行处理，其过程是比较两个个体的适应度函数值的大小，从中选取函数值大的个体采用变异操作和交叉操作生成新的个体 $\kappa_{i,j}^{t+1}$，然后与上一代的 $\kappa_{i,j}^t$ 进行比较，如果前者大于后者，则直接进入下一代，否则保持 $\kappa_{i,j}^t$ 不变。

3.4　边界处理

文献[18]中描述了在 CSO 算法中鸡位置容易出界的问题，本文认为小鸡虽然是跟随母鸡在其周围，但无法保证母鸡移动的位置接近边界的时候不会导致小鸡也会同样的超越边界的情况，这样容易导致算法在搜索的过程中陷入局部最优，从而引起

效率降低,因此采用边界变异的方法将越过界的小鸡位置转移到可行解的某个位置。设定随机数 r 为(0.2,1.0),如果小鸡超过可行解的上边界 U ,则将小鸡的位置调整为式(18),当小鸡的位置超过可行解的下边界 L 的时候,根据式(19)进行位置调整。

$$x_{i,j}=U-r(U-L) \tag{18}$$

$$x_{i,j}=L+r(U-L) \tag{19}$$

3.5 算法步骤

a)将 ICSO 算法中的个体与云计算中的资源调度方案一一对应。

b)初始化,设置最大迭代次数,CSO 算法相关参数初始值,设定 ω_{max} 和 $\omega_{\min}$ 分别为 0.9 和 0.1, $\lambda_{\max}$ 和 $\lambda_{\min}$ 值分别为 0.8 和 0.1, r 为 0.5。

c)采用式(9)~（11)对种群采用反向学习,优化初始种群。

d)对小鸡的位置采用式(13)~(15)进行更新,并对可能超越的边界使用式(18)（19)进行处理,提高整个算法的个体位置的质量;

e)采用式(15)（17)对鸡群算法的最优个体位置进行选择;

f）当迭代次数小于最大迭代次数,转步骤 d)继续执行,否则执行步骤 g);

g)位置最佳的个体就是最优个体即对应着云计算中的资源调度方案。

4 仿真实验

为了进一步说明 ICSO 能够有效地提高云计算中资源调度的效率,选取了几种经典的智能算法 GA、IPSO[19]、CSO 与 ICSO 在云计算资源调度下进行对比。硬件环境为酷睿 i3-8100,内存为 4 GB DDR4,硬盘为 500 GB,操作系统采用 Windows 7,通过 CloudSim 仿真平台下来对比资源调度的效果。实验中涉及到的算法相关参数如表 1 所示。本文从两个方面来说明 ICSO 算法具有的效果,一方面,将任务分为小规模任务和大规模任务两个方面分别从时间、花费和能耗三个指标进行对比,另一方面从负载均衡的效果方面进行对比。

表 1 对比算法的相关参数

参数	取值	说明
τ	0.0005	信息素
ρ_i	0.01	信息素发挥系数
p	0.5	路径选择概率
w	0.5	惯性权重
$c1$	0.5	粒子群的学习因子
$c2$	0.5	粒子群的学习因子
F	0.5	跟随数
$rand$	0.5	随机数
ω_{max}	0.9	最大权重值
$\omega_{\min}$	0.1	最小权重值
$\lambda_{\max}$	0.8	ICSO 学习因子最大值
$\lambda_{\min}$	0.1	ICSO 学习因子最小值
r	0.5	边界随机因子
Max	100	迭代次数

4.1 任务规模对比

4.1.1 小规模任务的性能对比

图 1~3 显示了在小规模任务下的四种算法在完成时间、花费和能耗方面进行的对比效果。图 1 中展示了四种算法相互之间竞争用户资源的概率较小,发生的冲突也较小,因此在完成时间方面几乎相差不大,ICSO 和 CSO 的算法优于 IPSO 算法和 ACO 算法,图 2 中展示了四种算法在花费方面的比较,从中可以发现 ICSO 算法的花费要小于其他的三种算法,这说明 ISCO 能够提高算法的性能,从而有效的降低花费。图 3 中展示了四种算法在能量消耗方面的比较,ICSO 算法在开始阶段消耗大,主要是因为初始化中采用了反向学习,随着任务数逐渐增大,ICSO 算法的曲线波动较小,而其他的三种算法曲线波动性较大,因此说明了 ICSO 在能量消耗具有较好的稳定性。

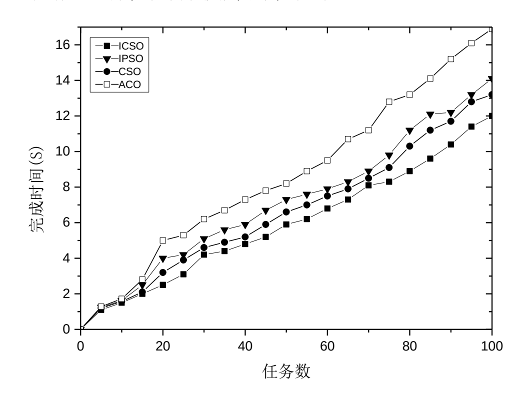


图 1 小规模任务下的四种算法完成时间对比

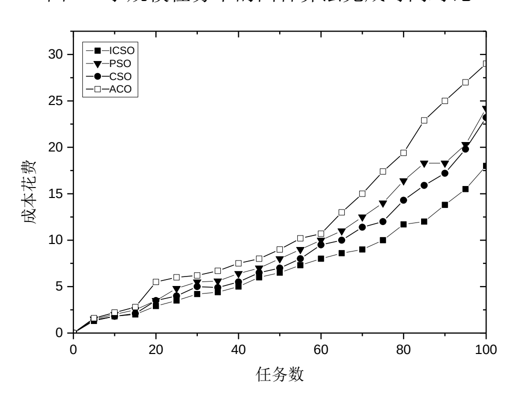


图 2 小规模任务下的四种算法成本花费对比

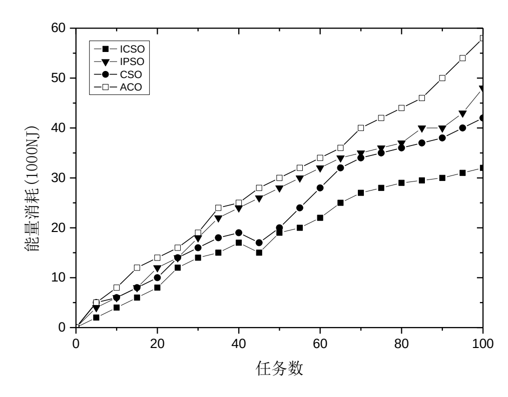


图 3 小规模任务下的四种算法能耗对比

4.1.2 大规模任务的性能对比

图 4-6 显示了在大规模任务下的四种算法在完成时间、花

费和能耗方面进行的对比效果。图 4 中展示了在大规模任务下的完成时间上,ICSO 算法具有明显的优势,随着任务数量的逐渐增多,四种算法的上升曲线形状是明显的,但是 ICSO 算法的上升是最慢的,这说明在完成时间上 ICSO 算法表现良好。图 5 中展示了在成本花费方面,ICSO 算法也具有明显的优势,通过对算法自身的改进,降低了成本的消耗,提高了算法的性能,节约了成本,降低了消耗。图 6 展示了四种算法在能耗方面的对比,伴随着任务数量的增多,ICSO 算法的曲线相比其他三种算法比较平缓,波动性小,而 CSO 算法,IPSO 算法和 ACO 算法曲线上升幅度大,这说明在能耗方面消耗较大,因此,ICSO 比较适应大规模下的任务调度。

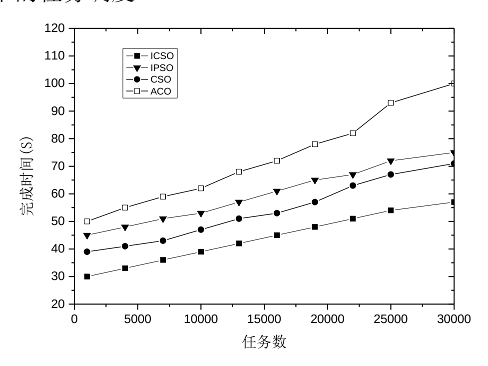


图 4　大规模任务下的四种算法完成时间对比

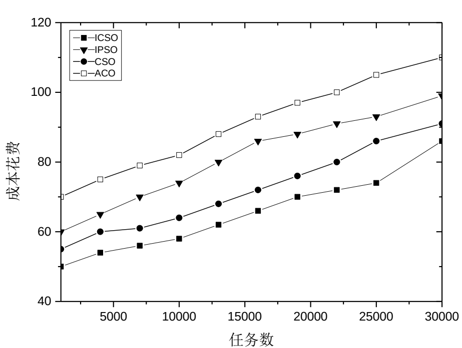


图 5　大规模任务下四种算法成本花费对比

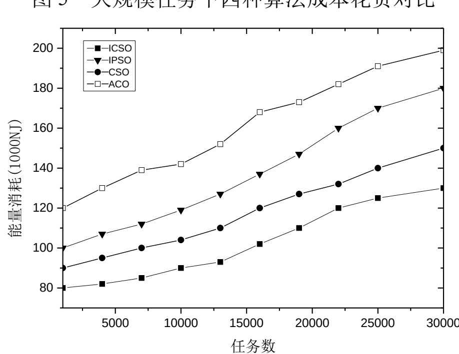


图 6　大规模任务下四种算法完成能耗对比

4.2　负载均衡对比

设定 10 000 个任务分布在五个资源点中{s1,s2,s3,s4,s5},其处理能力为{100,200,300,400,500},四种算法的对比效果如图 7 所示,由于云计算资源点处理能力不同,它们的负载都不相同,ICSO 算法负载比较均衡,这说明提高了云计算资源分配效果明显,而 CSO,IPSO 和 ACO 算法负载产生的数值不同,导致处理能力较强的资源点获得任务少,处理能力差的情况。

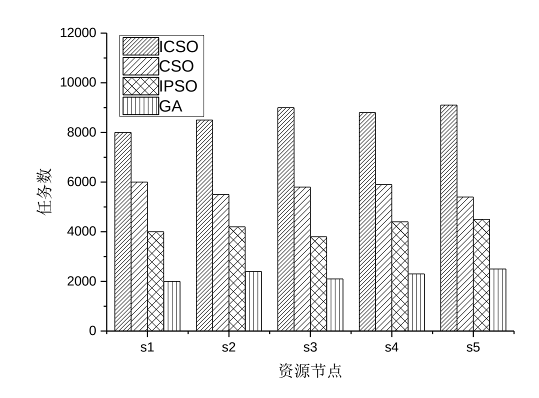


图 7　四种算法负载均衡对比

5　结束语

针对云计算资源分配的问题引入了 CSO 算法,并对其不足从 4 个方面进行了改进,将改进后的算法用于云计算下的资源调度中,仿真实验说明了 ICSO 算法能够适应云计算下资源调度并取得了比较好的效果,但由于本文仅仅是对几个常用的指标进行了描述,而忽视了云计算中的虚拟机在任务调度中的作用,将在下一步的研究中继续展开。

参考文献:

[1] 林伟伟, 齐德昱. 云计算资源调度研究综述 [J]. 计算机科学, 2012, 39 (10): 1-6 (Lin Weiwei, Qi Deyu. Survey of Resource Scheduling in Cloud Computing [J]. Computer Science, 2012. 39 (10): 1-6.)

[2] Shahdi S. New approach based on group technology for the consolidation problem in cloud computingmathematical model and genetic algorithm [J]. Computational and Applied Mathematics, 2016, DOI: 10. 1007//s40314-016-0362-4.

[3] Kềo D. Cloud computing-based parallel genetic algorithm for gene selection in cancer classification [J]. Neural Comput & Applic, 2016, DOI//10. 1007//s00521-016-2780-z

[4] Masdan A. A survey of pso-based scheduling algorithms in cloud computing [J]. Journal of Network and Systems Management, 2017, 25 (1): 122-158.

[5] Kumar N. Resource management using ANN-PSO techniques in cloud environment [C]// Proc of the International Congress on Information and Communication Technology. 2016: 419-428

[6] Kushwah V S. A basic simulation of ACO algorithm under cloud computing for fault tolerant [C]// Proc of International Conference on Data Engineering and Communication Technology, 2017: 465-472.

[7] Ragmani A. A performed load balancing algorithm for public Cloud computing using ant colony optimization [C]// Proc of the 2nd International Conference on Cloud Computing Technologies and Applications. 2016.

[8] 孔飞, 吴定会. 一种改进的鸡群算法 [J]. 江南大学学报: 自然科学版, 2015, 14 (6): 681-688 (Kong Fei, Wu Dinghui. An improved chicken swarm optimization algorithm [J] , Journal of Jiangnan University: Natural Science Edition, 2015, 14 (6): 681-688.)

[9] 胡汉梅, 李静雅, 黄景光. 基于鸡群算法的微网经济运行优化 [J]. 高

压电器, 2017, 53 (1): 119-125. (Hu Hanmei, Li Jingya, Huang Jingguang. Economic operation optimization of micro-grid based on chicken swarm optimization algorithm [J]. High Voltage Apparatus, 2017, 53 (1): 119-125.)

[10] 许世鹏, 吴定会, 孔飞. 基于改进鸡群算法的柔性作业车间调度问题求解 [J]. 系统仿真学报, 2017, 29 (7): 1497-1505 (Xu Shipeng, Wu Dinghui, Kongfei. Solving flexible Job-Shop scheduling problem by improved chicken swarm optimization algorithm [J]. Journal of System Simulation, 2017, 29 (7): 1497-1505)

[11] 吴定会, 许世鹏. Pareto 熵鸡群算法求解多目标柔性作业车间调度问题 [J]. 小型微型计算机系统, 2017, 38 (12): 2683-2688. (Wu Dinghui, Xu Shipeng. Solving multi-objective flexible Job Shop scheduling problem by the chicken swarm optimization algorithm based on Pareto entropy [J]. Journal of Chinese Computer Systems, 2017, 38 (12): 2683-2688.)

[12] 洪杨, 于凤芹. 改进的鸡群算法并用于多分类器系数优化 [J]. 计算机工程与应用, 2017, 53 (9): 158-161. (Hong Yang, Yu Fengqin. Improved chicken swarm optimization and its application in coefficients optimization of multi-classifier [J]. Computer Engineering and Applications, 2017, 53 (9): 158-161.)

[13] 魏月梅, 池丽敏. 耗散鸡群算法在水库优化调度中的应用 [J]. 水力发电，2017, 43 (3): 111-114. (Wei Yuemei, Chi Liming. Application of dissipation chicken swarm optimization in reservoir optimal operation [J]. Water Power, 2017, 43 (3): 111-114.)

[14] 许仪勋, 李旺, 李东东, 等. 基于改进鸡群算法的非侵入式家电负荷分解 [J]. 电力系统保护与控制, 2016, 44 (13): 27-32. (Xu Yixun, Li Wang, LiDongdong. Disaggregation for non-invasive domestic appliances based on the improved chicken swarm optimization algorithm [J]. Power System Protection and Control, 2016, 44 (13): 27-32.)

[15] Tizhoosh H R. Opposition-based learning: a new sheme for machine intelligence [C]// Proc of International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelliget Agents, Web Technologies and Internet Commerce. Washington DC: IEEE Computer Society, 2005: 695-701.

[16] Wang Hui, Wu Zhijian, Rahnamayan S, *et al.* Enhancing particle swarm optimization using generalized oppsiton-based learning [J]. Informtion Sciences, 2011, 181 (20): 4699-3714.

[17] 白艳敏. 粒子群算法及其应用研究 [D]. 兰州: 兰州交通大学, 2013: 8-9 (Bai Yanmin. Particle swarm optimization and its application [D]. Lanzhou: Lanzhou Jiaotong University, 2013: 8-9.)

[18] 杨菊婧, 张达敏, 张慕雪. 一种混合改进的鸡群优化算法 [J/OL]. 2018, 35 (11) . [2017-11-10]. http://www. arocmag. com/article/02-2018-11-004. html (Yang Juting, Zhang Damin, Zhang Muxue. Hybrid improved for chicken swarm optimization algorithm [J]. Application Research of Computers, 2018, 35 (11) .)

[19] 王月, 刘亚秋, 郭继峰. 基于离散粒子群优化的云计算 QoS 调度算法 [J]. 计算机工程, 2017, 43 (6): 111-117. (Wang Yue, Liu Yaqiu, Guo Jifeng. QoS scheduling algorithm in cloud computing based on discrete particle swarm optimization [J]. Computer Engineering, 2017, 43 (6): 111-117.)